



成都远向电子科技有限公司产品说明书

产品型号：YX-DID0-RS485-008

全部资料下载地址：<http://ask.zstel.com:8090>

技术支持服务电话：028-64267900

技术支持专员企业 QQ：3183329475

官网网站：<https://www.zstel.com/>

硬件/软件技术定制热线：19150158475 张工

目录

一、 产品概述.....	3
1.1 概述.....	3
1.2 性能特点.....	3
1.3 技术参数.....	3
二、 外观尺寸.....	4
2.1 产品外观.....	4
2.2 产品尺寸图.....	4
三、 产品接线图、跳线、指示灯说明.....	6
3.1 端子接口.....	6
3.2 接线图.....	6
3.3 LED 指示灯.....	7
四、 ModbusRTU 通讯协议地址以及案例说明.....	8
4.1 通讯协议.....	8
4.2 寄存器地址.....	8
4.3 Modbus RTU 功能码.....	8
4.4 Modbus 通讯实例.....	8
五、 软件操作.....	17
5.1 配置软件.....	17
5.2 配置基本参数.....	18
5.3 DO 继电器输出相关参数.....	18
5.3 DI 开关量输入相关参数.....	18
5.6 其他功能.....	19
六、 协议详解.....	28
6.1 功能码描述.....	28
6.2 错误码描述.....	34
6.3 CRC 校验算法.....	34
七、 更改记录.....	36

一、产品概述

1.1 概述

YX-DID0-RS485-008 是一款工业级标准开关量采集产品，共有 8 个开关量输入通道。每个通道均可以分别设置联动控制 DO 继电器输出；RS-485 通讯接口使用标准 Modbus RTU 协议，符合工业标准。

1.2 性能特点

- 防死机软、硬件看门狗
- 5~35V 带防反接、过压过流保护电源
- 8 路继电器输出
- 8 路带光耦隔离开关量输入
- 可配置高、低输入
- 高性能低功耗 32 位 ARM 嵌入式 CPU
- 支持 ModbusRTU 从站协议
- 支持脚本编程，自定义控制逻辑
- 多路指示灯
- 带防雷、静电保护 RS485 通讯接口
- 工业机温度范围，应对严苛现场环境

1.3 技术参数

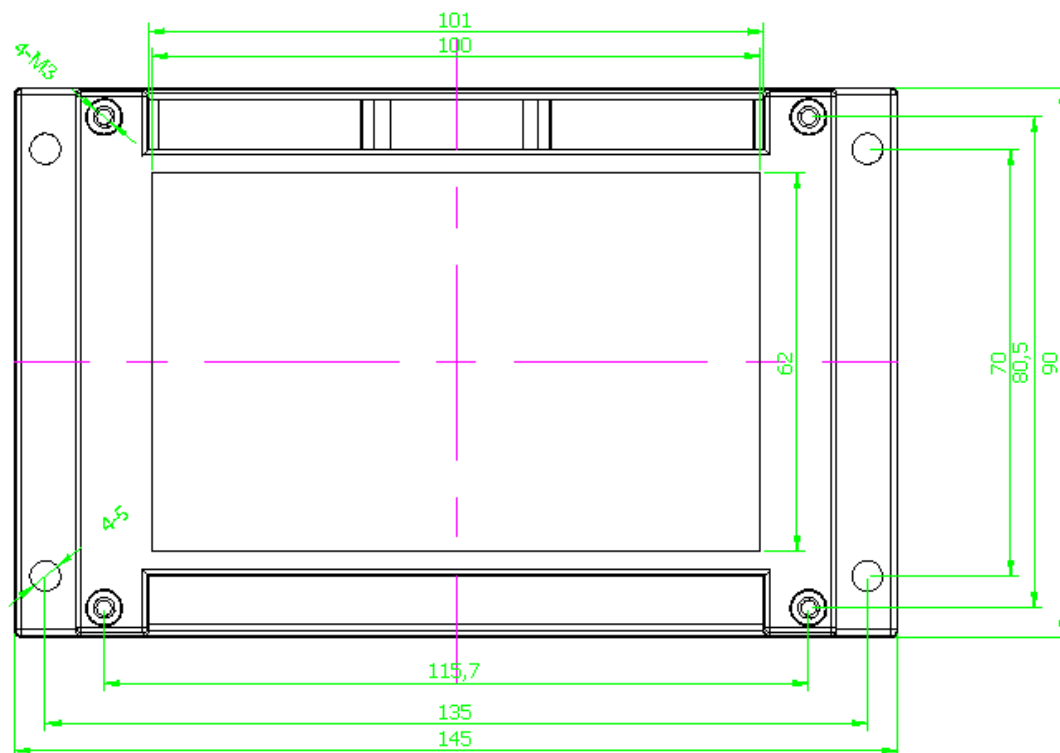
开关量输入	DI	8 路单端
	DI 响应	下降沿，低输入
	响应时间	30ms
	输入类型	干节点、无源输入、有源（NPN）输入
	最大输入电压	电源输入电压
	最大输入电流	20mA
	输入阻抗	2.7kΩ
开关量输出	继电器类型	8 路继电器 常开
	触点电阻	100mΩ
	机械寿命	1x10 ⁶ 次
	最大切换电压	0~250VAC，0~30VDC
	最大切换电流	0~5A
通讯接口	通讯接口	RS485
	波特率	1200~115200bps
	数据格式	8N1, 8E1, 801, 8N2, 8E2, 802
	通讯协议	ModbusRTU
电源参数	电源规格	DC 5~36V
	功耗	12V-6W
工作环境	工作温度、湿度	-40~85℃，0%~95%RH
其他	尺寸	82*50*32

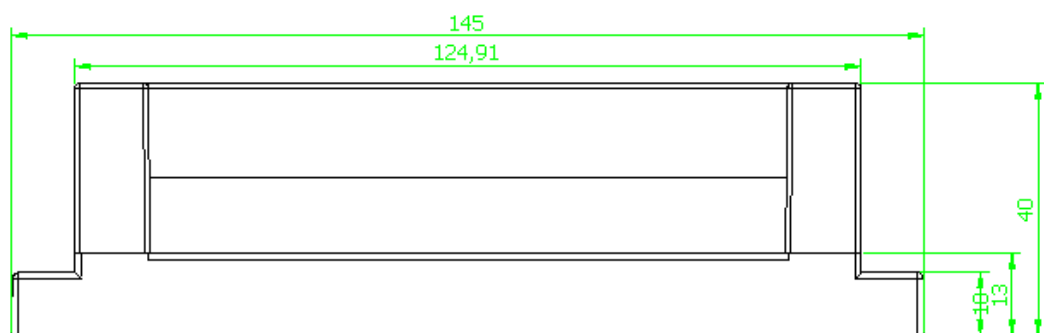
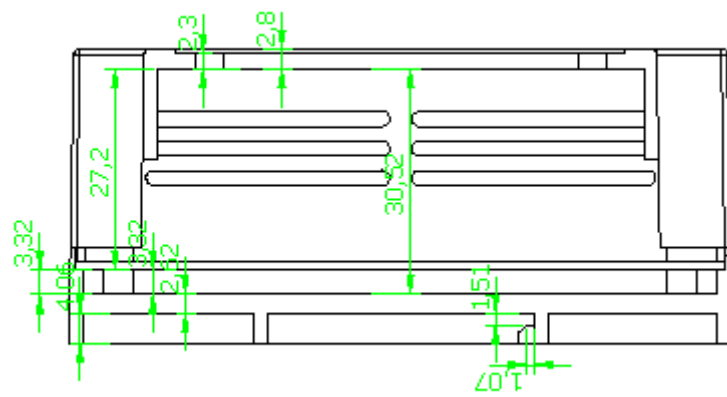
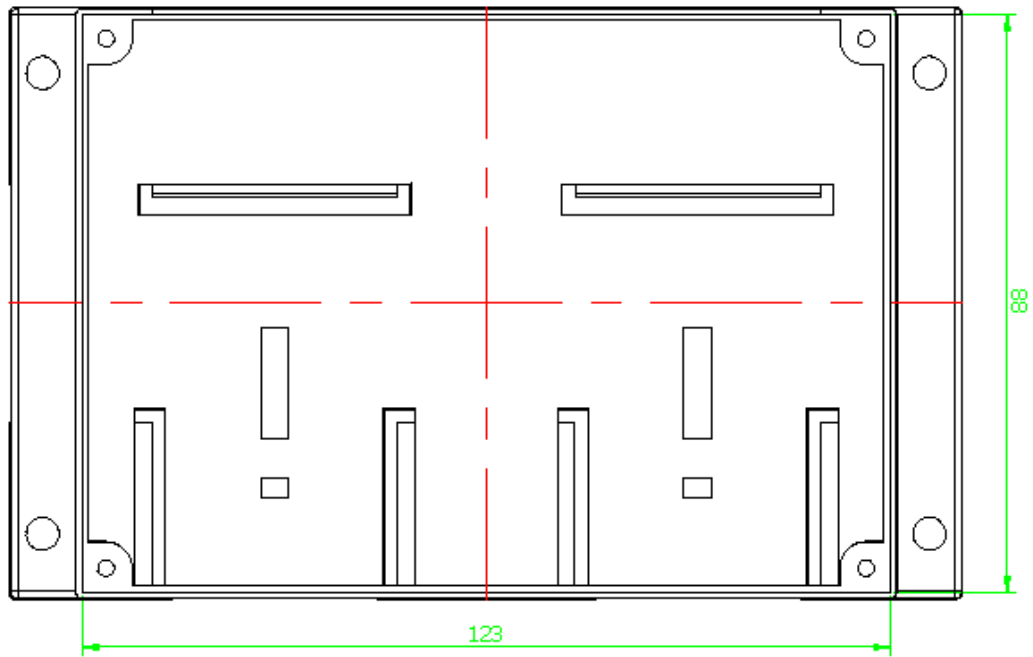
二、外观尺寸

2.1 产品外观



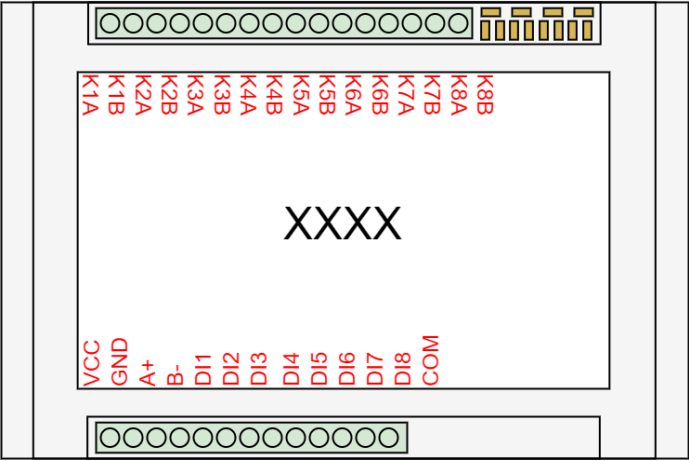
2.2 产品尺寸图





三、产品接线图、跳线、指示灯说明

3.1 端子接口



● 顶部 16 槽接线位：

- | | |
|----------------|----------------|
| K1A: 继电器 1 进线端 | K5A: 继电器 5 进线端 |
| K1B: 继电器 1 出线端 | K5B: 继电器 5 出线端 |
| K2A: 继电器 2 进线端 | K6A: 继电器 6 进线端 |
| K2B: 继电器 2 出线端 | K6B: 继电器 6 出线端 |
| K3A: 继电器 3 进线端 | K7A: 继电器 7 进线端 |
| K3B: 继电器 3 出线端 | K7B: 继电器 7 出线端 |
| K4A: 继电器 4 进线端 | K8A: 继电器 8 进线端 |
| K4B: 继电器 4 出线端 | K8B: 继电器 8 出线端 |

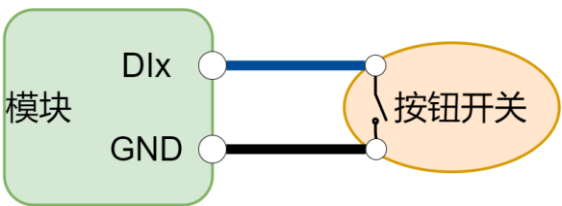
● 底部 13 槽接线位：

- | | |
|-----------------|----------------|
| VCC: 电源正极 | DI4: 开关量输入通道 4 |
| GND: 电源负极 | DI5: 开关量输入通道 5 |
| A+: RS485 通讯线 A | DI6: 开关量输入通道 6 |
| B-: RS485 通讯线 B | DI7: 开关量输入通道 7 |
| DI1: 开关量输入通道 1 | DI8: 开关量输入通道 8 |
| DI2: 开关量输入通道 2 | COM: 开关量输入公共端 |
| DI3: 开关量输入通道 3 | |

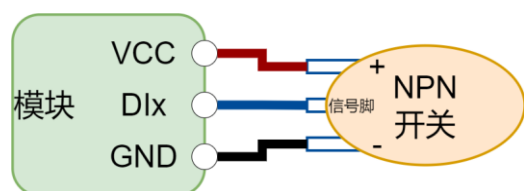
3.2 接线图

(1) 开关量输入（DI）接线图

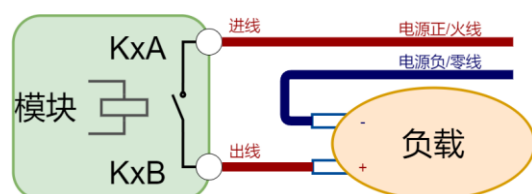
无源按钮开关



有源 NPN 开关

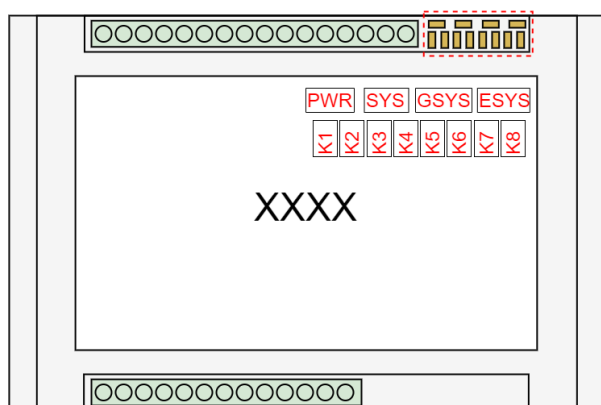


(2) 开关量输出 (DO) 接线图



注意：DO 输出为继电器输出，输出负载参数见上文“技术参数”

3.3 LED 指示灯



12 个 LED 指示灯：

- **PWR**：电源指示灯，电源正常工作时常亮
- **SYS**：系统状态灯，正常运行时每秒闪烁一次
- **GSYS**：闲置，暂时未使用
- **ESYS**：闲置，暂时未使用
- **K1**：输出通道 D01 状态指示灯。亮：输出信号、灭：无输出
- **K2**：输出通道 D02 状态指示灯。亮：输出信号、灭：无输出
- **K3**：输出通道 D03 状态指示灯。亮：输出信号、灭：无输出
- **K4**：输出通道 D04 状态指示灯。亮：输出信号、灭：无输出
- **K5**：输出通道 D05 状态指示灯。亮：输出信号、灭：无输出
- **K6**：输出通道 D06 状态指示灯。亮：输出信号、灭：无输出
- **K7**：输出通道 D07 状态指示灯。亮：输出信号、灭：无输出
- **K8**：输出通道 D08 状态指示灯。亮：输出信号、灭：无输出

四、ModbusRTU 通讯协议地址以及案例说明

4.1 通讯协议

本产品支持标准 Modbus RTU 从站协议，能够支持标准 Modbus RTU 组态软件，详细介绍参考本文第六章内容

4.2 寄存器地址

寄存器地址	名称	字节数	说明	备注
数字量输入				
0x0010（16）	DI1	2	开关量 DI 输入通道 1	0000 表示无输入 0001 表示有输入
0x0011（17）	DI2	2	开关量 DI 输入通道 2	
0x0012（18）	DI3	2	开关量 DI 输入通道 3	
0x0013（19）	DI4	2	开关量 DI 输入通道 4	
0x0014（20）	DI5	2	开关量 DI 输入通道 5	
0x0015（21）	DI6	2	开关量 DI 输入通道 6	
0x0016（22）	DI7	2	开关量 DI 输入通道 7	
0x0017（23）	DI8	2	开关量 DI 输入通道 8	
数字量输出				
0x00018（24）	D01	2	数字量 D0 输出通道 1	0000 表示断开 0001 表示闭合
0x00019（25）	D02	2	数字量 D0 输出通道 2	
0x0001A（26）	D03	2	数字量 D0 输出通道 3	
0x0001B（27）	D04	2	数字量 D0 输出通道 4	
0x0001C（28）	D05	2	数字量 D0 输出通道 5	
0x0001D（29）	D06	2	数字量 D0 输出通道 6	
0x0001E（30）	D07	2	数字量 D0 输出通道 7	
0x0001F（31）	D08	2	数字量 D0 输出通道 8	

4.3 Modbus RTU 功能码

功能码	操作	说明
01	读取单位 D0 状态	Bit 位表示 D0 输出状态
03	读取 DI, D0 寄存器值	读取 DI, D0 寄存器值
04	读取 DI, D0 寄存器值	读取 DI, D0 寄存器值
05	写单个 D0	0xFF00: 闭合;0x0000: 断开
06	写单个 D0	0x0001: 闭合;0x0000: 断开
0F	写多个 D0	参照第六章内容
10	写多个 D0	参照第六章内容

详细讲解参照本文第六章内容

4.4 Modbus 通讯实例

- (1) 读取 D0:
- a. 用 01 功能码读取 D0

发送: 01 01 00 18 00 08 BD CB

接受: 01 01 01 00 51 88

The image shows two software windows. The top window is MODSCAN32, a Modbus master simulator. It displays configuration for Device ID 1, Address 0025, and Length 8. The MODBUS Point Type is set to 01: COIL STATUS. It shows 539 polls and 513 valid slave responses. A list of coil addresses (00025 to 00028) and their values (<0>) is visible. The bottom window is Modbus Poll - Mbpoll1, showing a table of coil addresses and their values. The table has columns for Address, Alias, and Value. The values are all 0.

MODSCAN32 For OPTO22 - 工控技术交流QQ群: 207149229 - 无标题

文件(F) 连接设置(C) 配置(S) 查看(V) 窗口(W) 帮助(H)

无标题

Address: 0025 Device Id: 1 Number of Polls: 539
Length: 8 MODBUS Point Type: 01: COIL STATUS Valid Slave Responses: 513
Reset Ctrs

00025: <0> 00029: <0>
00026: <0> 00030: <0>
00027: <0> 00031: <0>
00028: <0> 00032: <0>

ModScan32 - (COMM7) Polls: 539 Resps: 513

Modbus Poll - Mbpoll1

File Edit Connection Setup Functions Display View Window Help

05 06 15 16 17 22 23 TC ? ?

Mbpoll1

Tx = 297: Err = 0: ID = 1: F = 04: SR = 1000ms

	Alias	00024
24		0
25		0
26		0
27		0
28		0
29		0
30		0
31		0

b. 用 03 功能码读取 D01

发送: 01 03 00 18 00 01 04 0D

接受: 01 03 02 00 00 B8 44

MODSCAN32 For OPTO22 - 工控技术交流QQ群: 207149229 - 无标题

文件(F) 连接设置(C) 配置(S) 查看(V) 窗口(W) 帮助(H)

无标题

Address: 0025 Device Id: 1 Number of Polls: 561
 Length: 8 MODBUS Point Type 03: HOLDING REGISTER Valid Slave Responses: 535
 Reset Ctrs

40025: <00000> 40029: <00000>
 40026: <00000> 40030: <00000>
 40027: <00000> 40031: <00000>
 40028: <00000> 40032: <00000>

ModScan32 - (COMM7) Polls: 561 Resps: 535

Modbus Poll - Mbpoll1

File Edit Connection Setup Functions Display View Window Help

05 06 15 16 17 22 23 TC TC ? ?

Mbpoll1

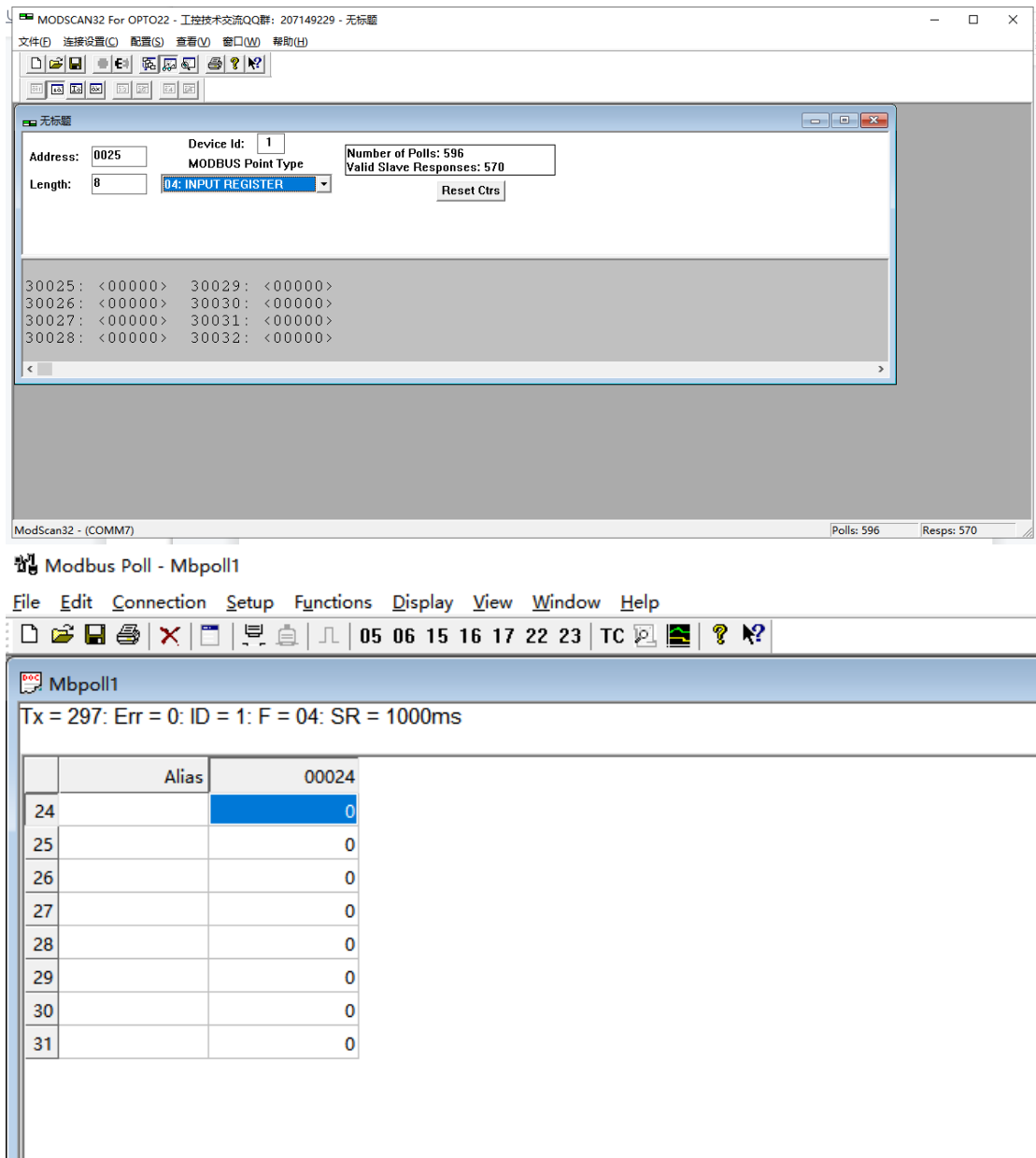
Tx = 297: Err = 0: ID = 1: F = 04: SR = 1000ms

	Alias	00024
24		0
25		0
26		0
27		0
28		0
29		0
30		0
31		0

c. 用 04 功能码读取 D01

发送: 01 04 00 18 00 01 B1 CD

接受: 01 04 02 00 00 B9 30

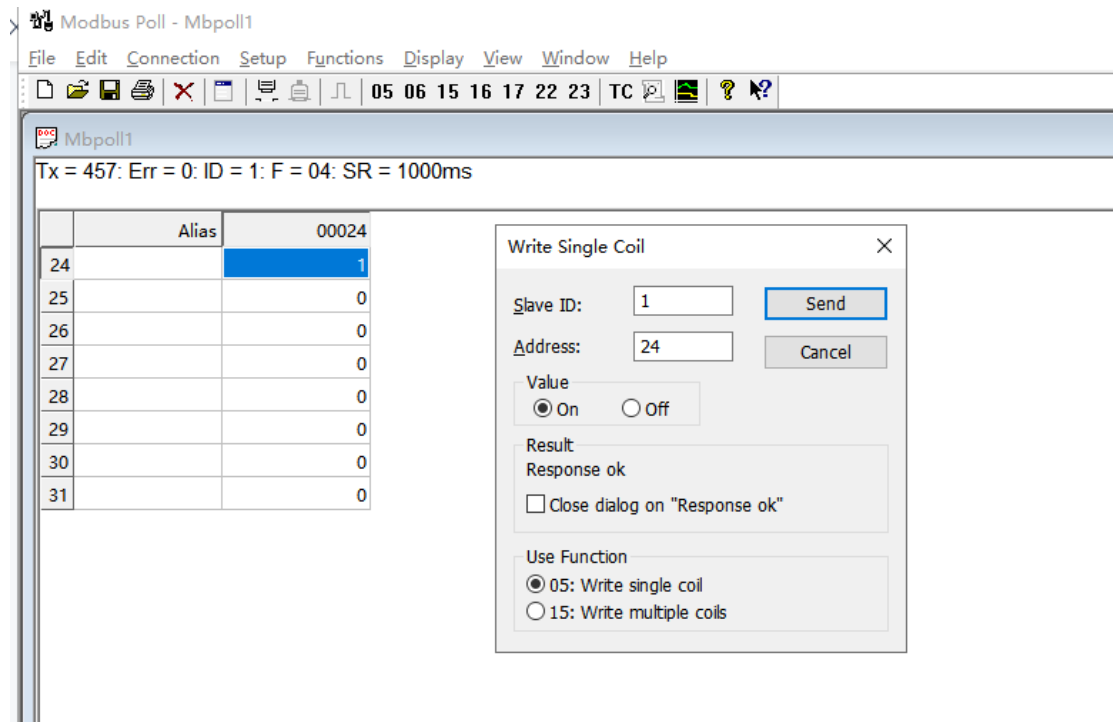


(2) 操作 D01:

a. 用 05 功能码操作单个 D01

发送: 01 05 00 18 FF 00 0C 3D

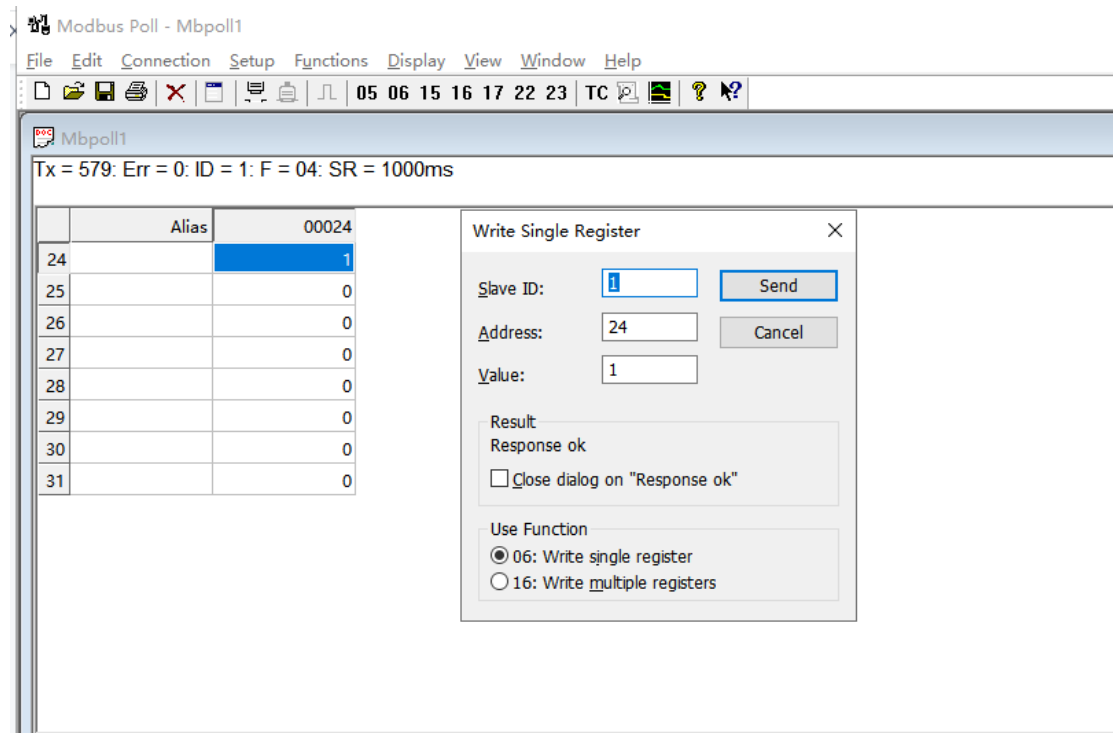
接受: 01 05 00 18 FF 00 0C 3D



b. 用 06 功能码操作单个 D01

发送: 01 06 00 18 00 01 C8 0D

接受: 01 06 00 18 00 01 C8 0D



c. 用 0F 功能码操作多个 D01、D02

发送: 01 0F 00 18 00 02 01 03 BE 94

接受: 01 0F 00 18 00 02 54 0D

Modbus Poll - Mbpoll1

File Edit Connection Setup Functions Display View Window Help

05 06 15 16 17 22 23 TC ? ?

Mbpoll1

Tx = 611: Err = 0: ID = 1: F = 04: SR = 1000ms

	Alias	00024
24		1
25		1
26		0
27		0
28		0
29		0
30		0
31		0

15: Write Multiple Coils

Slave ID: 1
Address: 24
Quantity: 8

☒ Coil 24
☒ Coil 25
☒ Coil 26
☐ Coil 27
☐ Coil 28
☐ Coil 29
☐ Coil 30
☐ Coil 31

Send
Cancel
Open
Save

d. 用 10 功能码操作多个 D01、D02

发送: 01 10 00 18 00 02 04 00 01 00 01 63 05

接受: 01 10 00 18 00 02 C1 CF

Modbus Poll - Mbpoll1

File Edit Connection Setup Functions Display View Window Help

05 06 15 16 17 22 23 TC ? ?

Mbpoll1

Tx = 678: Err = 0: ID = 1: F = 04: SR = 1000ms

	Alias	00024
24		1
25		1
26		0
27		0
28		0
29		0
30		0
31		0

16: Write Multiple Registers

Slave ID: 1
Address: 24
Quantity: 8
Type: Signed

024 = 1
025 = 1
026 = 0
027 = 0
028 = 0
029 = 0
030 = 0
031 = 0

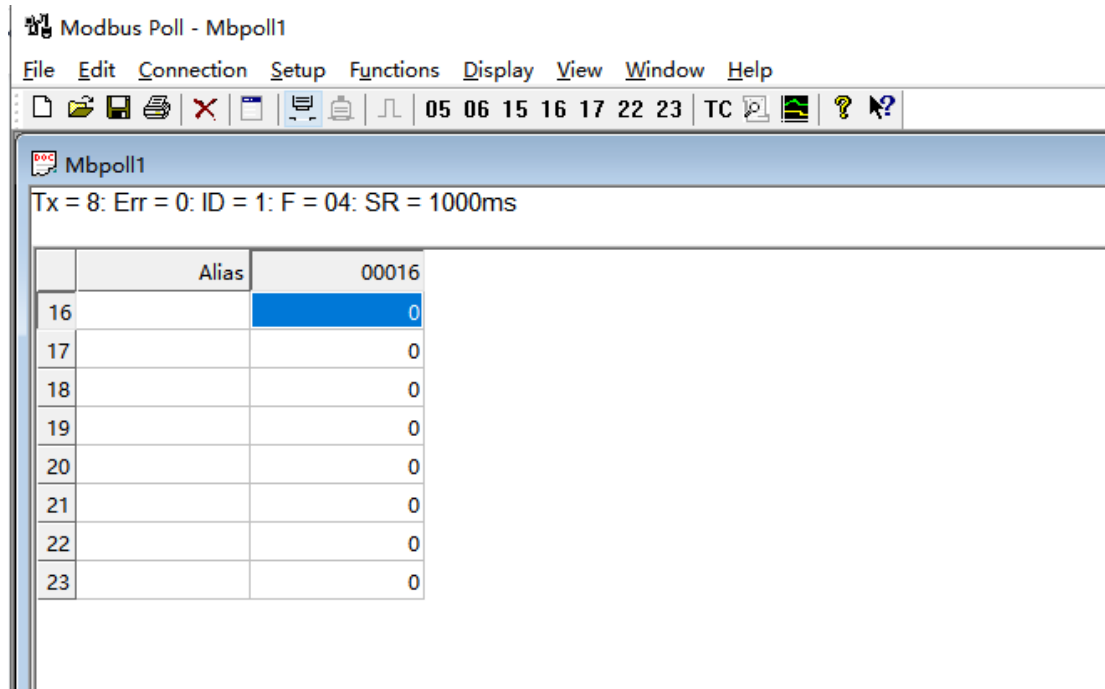
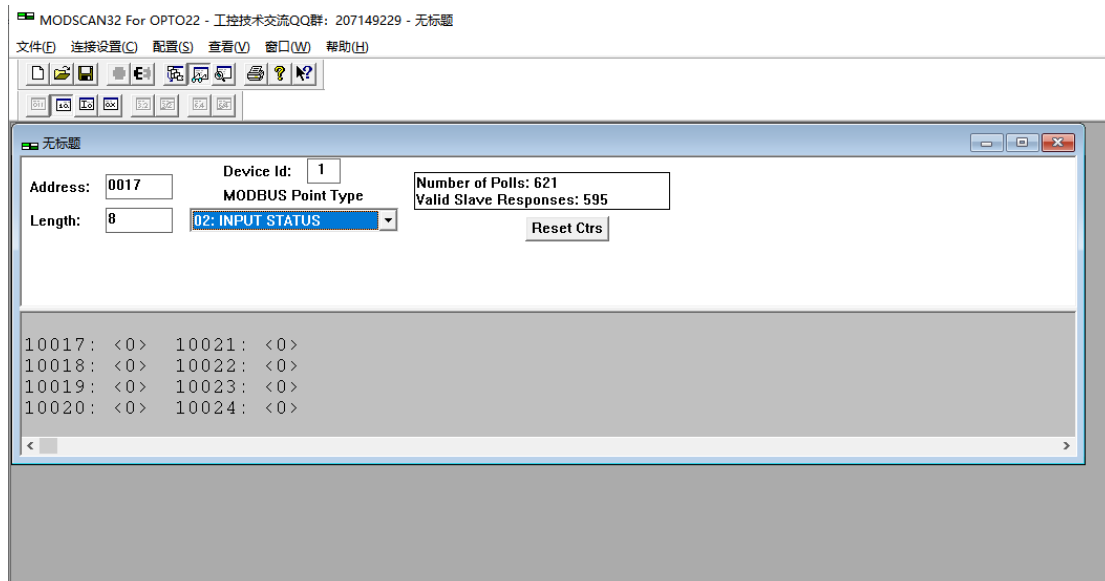
Send
Cancel
Edit
Open
Save

(3) 读取 DI:

a. 用 02 功能码读取 DI1-DI8 (DI1 到 DI8 处于“断开”状态)

发送: 01 02 00 10 00 08 85 CF

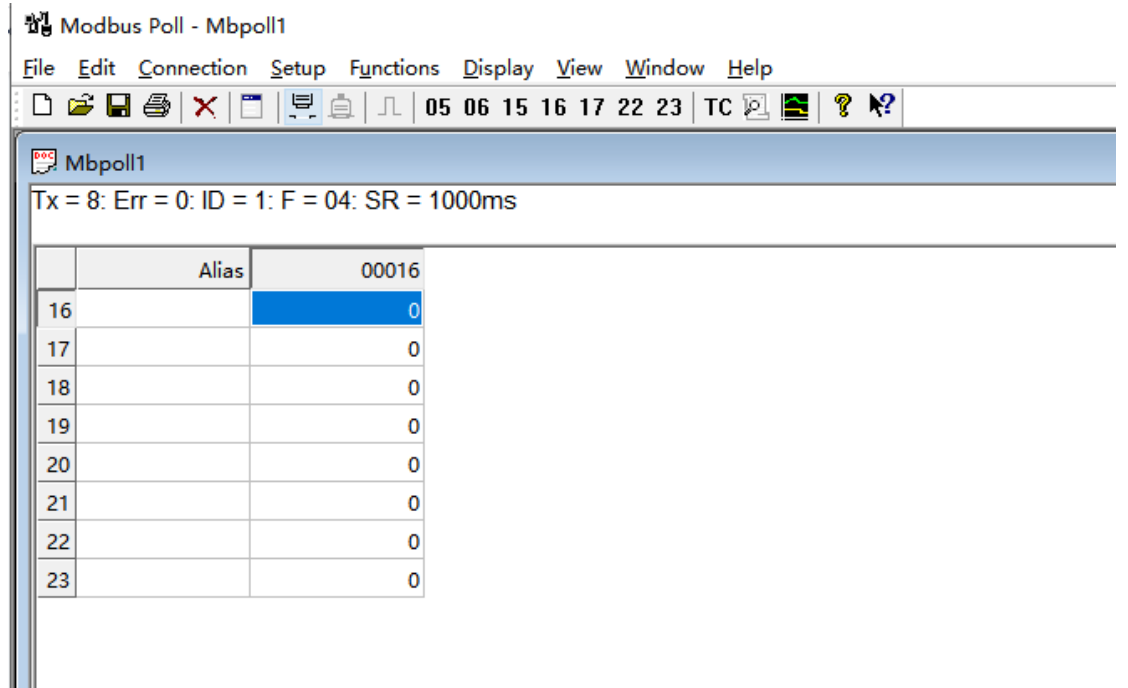
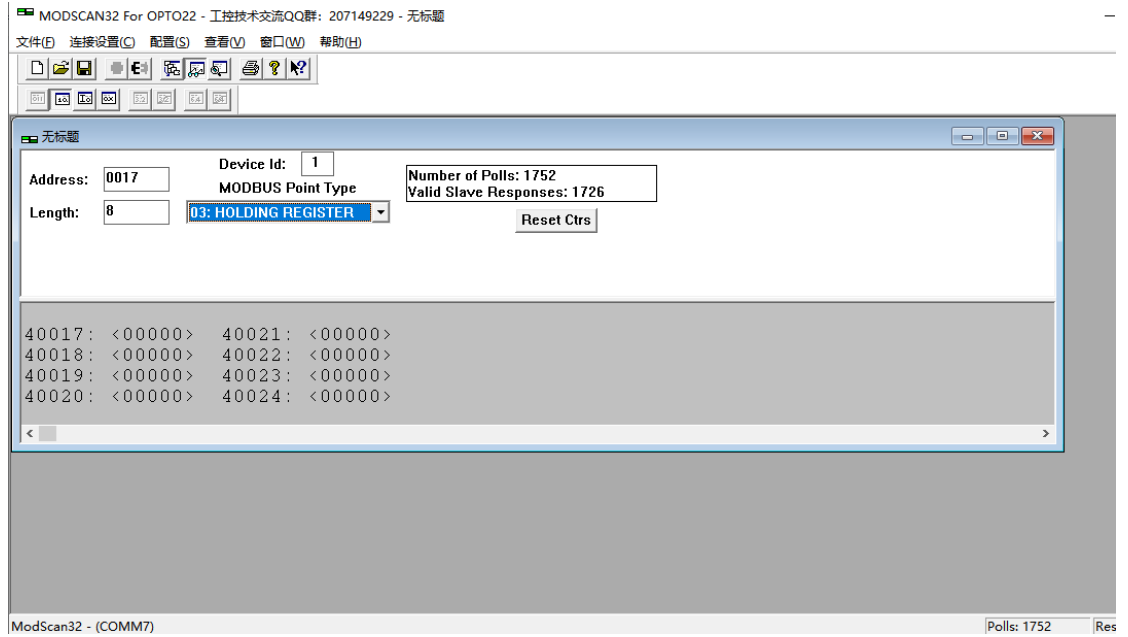
接受: 01 02 02 00 00 A1 88



b.用 03 功能码读取 DI1-DI8 (DI1 和 DI2 处于“断开”状态)

发送: 01 03 00 10 00 08 45 C9

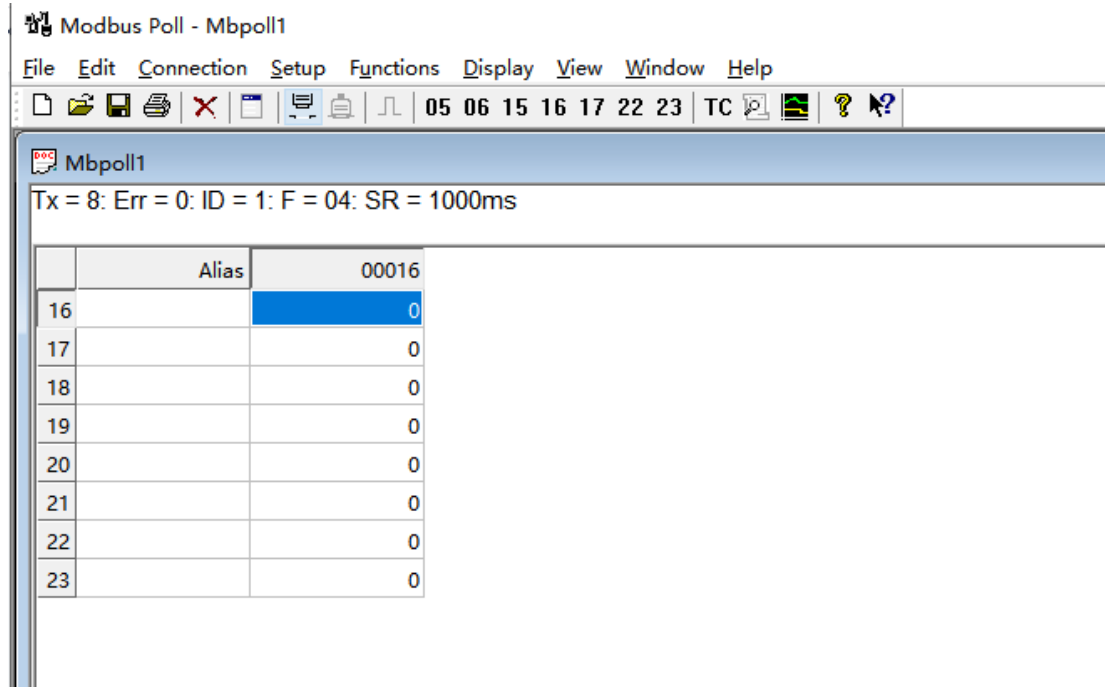
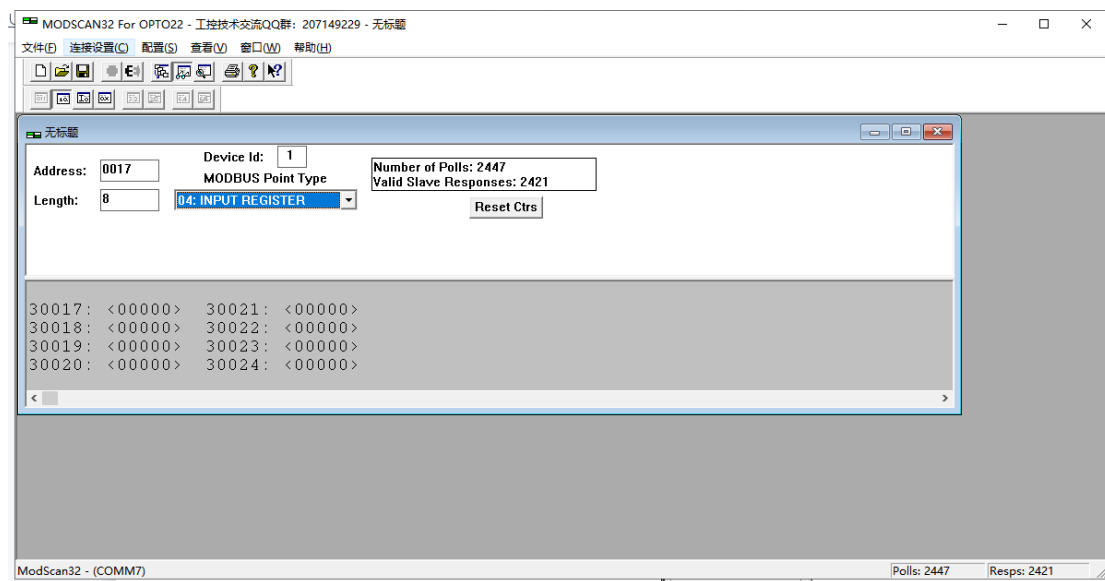
接受: 01 03 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 E4 59



c. 用 04 功能码读取 DI1-DI8（DI1 到 DI8 处于“断开”状态）

发送: 01 04 00 10 00 08 F0 09

接受: 01 04 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 2C



五、软件操作

设备参数配置教程，结合《用户测试文档》即可对设备进行简单测试

5.1 配置软件

参数配置软件介绍：



5.1.1 配置软件包含有：

- 功能区：包含有配置软件所支持功能, 以及功能那个切换选项
- 参数配置主区域：参数配置主要区域，参数项的读取、写入临时列表
- 串口/命令集区：涉及模块的参数读、写、重启等操作
- 串口日志区：命令集的操作日志

5.1.2 参数配置准备：

- (1) 用 USB-485 工具连接设备到电脑
- (2) 在串口配置框内配置串口波特率、停止位、校验位、数据位；（默认波特率 9600，数据位 8，停止位 1，校验位 None）
- (3) 选择串口配置框子项“命令集”



- (4) 点击“**读取参数**”命令按钮，读取设备参数（不同设备拥有不同指令集）
- (5) 双击对应参数项的“**参数值**”，然后对参数进行修改
- (6) 修改完参数后需要点击命令集里的“**设置参数**”，写入到模块中
- (7) 写入完成在日志区域会提示成功。



- (8) 通过点击“**重启设备**”按钮，重启模块设备使配置参数生效

5.2 配置基本参数

该系列参数涉及到对 485 通讯 Modbus 协议相关配置。

参数名称	参数值	参数说明
<基本参数>		
Modbus地址	双击修改参数	设备的Modbus地址, 1~255
通信模块波特率		与通信模块的波特率一致, 一般设置为9600
通信模块奇偶校验		与通信模块的奇偶校验一致, 一般设置为8N1

- **Modbus 地址**: Modbus 地址参数, 可设置 1~255
- **通讯模块波特率**: 设备 485 通讯波特率（波特率支持主流的波特率选项）
- **通讯模块就校验**: 设备 485 通讯奇偶校验位, 可配置 8N1, 8E1, 8O1...

5.3 D0 继电器输出相关参数

部分产品包含有多个 D0 输出或者不包含有 D0 输出功能，具体请根据实际配置软件显示栏目进行配置。D0 功能测试可以参考《[用户测试文档](#)》。

<D0继电器输出相关参数>		
DO1初始值	双击修改参数	定义第1路继电器的初始状态
DO2初始值		定义第2路继电器的初始状态

- **DOx 初始值**: 通过选择参数可配置开机上电后 D0 输出状态，可配置为：“等待上位操作”，“低（断开）”，“高（闭合）”，“记忆上次状态”，默认为“等待上位操作”状态。

5.3 DI 开关量输入相关参数

<DI开关量相关参数>		
DI1告警触发	双击修改参数	定义第1路开关量发生何种变化后设备发出告警信息
DI1告警控制		定义第1路开关量触发后控制DO输出状态切换

- **DIx 告警触发：**可配置高触发、低触发、不触发，主要用于联动控制。
- **DIx 告警控制：**联动控制 DO 功能，DI 告警事件触发后立即翻转 DO 输出状态（由“低/断开”切换为“高/闭合”状态）

5.6 其他功能

5.6.1 DI-DO 联动控制案例

（1）必要参数

<DO继电器输出相关参数>		
DO1初始值		定义第1路继电器的初始状态
<DI开关量相关参数>		
DI1告警触发		定义第1路开关量发生何种变化后设备发出告警信息
DI1告警控制		定义第1路开关量触发后控制DO输出状态切换

- DOx 初始值：配置正常闲置状态 DOx 状态
- DIx 告警触发：DIx 的告警触发状态（高触发、低触发）
- DIx 告警控制：DIx 告警事件触发后控制 DOx 项，只能一对一
- （如需一对多，请使用脚本编程功能实现）

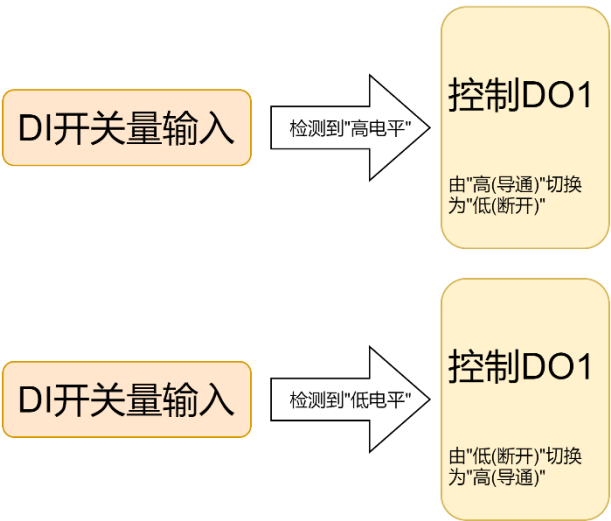
（2）实际案例

配置如下参数：

- DO1 初始值：低（断开）
- DI1 告警触发：高触发
- DI1 告警控制：DO1

参数名称	参数值	参数说明
<DO继电器输出相关参数>		
DO1初始值	低（断开）	定义第1路继电器的初始状态
<DI开关量相关参数>		
DI1告警触发	高触发	定义第1路开关量发生何种变化后设备发出告警信息
DI1告警控制	DO1	定义第1路开关量触发后控制DO输出状态切换

告警触发后流程如下图：



5.6.2 消息告警内容参数配置(无线版)

(1) 必要参数

注：需要配合本公司远程模块, 485 有线版不开放该功能

- 安装地址： 主要用于报警通知使用（设备需包含有短信模块）
- 设备身份 ID： 主要用于报警通知使用（设备需包含有短信模块）
- DIx 告警周期：用于循环发送告警消息时间
- DIx 告警短信内容：告警发送消息
- DIx 恢复消息内容：消除告警状态发送消息内容

<DI告警短信相关参数>		
安装地址		最长16个汉字,32个字符, 短信报警用
设备身份ID		8位编码, 短信报警时有效
DI1告警周期		循环告警的时间间隔, 单位为分钟, 0表示只告警一次
DI1告警短信内容		最多30个汉字或60个字符
DI1恢复短信内容		最多30个汉字或60个字符

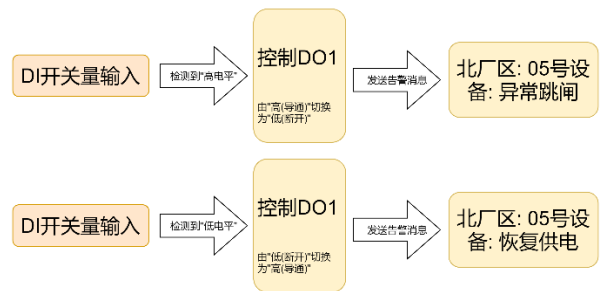
(2) 告警内容案例：

配置如下：

DO1 初始值： “高(闭合)” 设备身份 ID： “05 号设备： ”
DI1 告警触发： “高电平” DI1 告警周期： “1”
DI1 告警控制： “DO1” DI1 告警内容： “异常跳闸”
安装地址： “北厂区： ” DI1 恢复内容： “恢复供电”

参数名称	参数值	参数说明
<DO继电器输出相关参数>		
DO1初始值	低 (断开)	定义第1路继电器的初始状态
<DI开关量相关参数>		
DI1告警触发	高触发	定义第1路开关量发生何种变化后设备发出告警信息
DI1告警控制	DO1	定义第1路开关量触发后控制DO输出状态切换
<DI告警短信相关参数>		
安装地址	北厂区	最长16个汉字,32个字符, 短信报警用
设备身份ID	05号设备	8位编码, 短信报警时有效
DI1告警周期	1	循环告警的时间间隔, 单位为分钟, 0表示只告警一次
DI1告警短信内容	异常跳闸	最多30个汉字或60个字符
DI1恢复短信内容	恢复供电	最多30个汉字或60个字符

告警触发流程入下图：



若只需要联动功能只需配置 “DOx 初值” 和 “DIx 告警触发” 以及 “DIx 告警控制” 即可（消息内容为空则不发送内容）

5.6.3 RTU 数据可视化测试

参数配置软件包含有 RTU 数据可视化功能, 可实现简单的模块功能测试。
操作步骤如下图:



- 选择“RTU 可视化”界面
- 配置 USB 转 485 通讯参数并打开串口
- 设置 Modbus 子站地址为默认“1”
- 点选“自动读取”复选框

注意：此时界面左侧仪表盘会根据模拟量信号源的输出显示对应电流读数，右侧会有上下行通讯报文。若无上行数据请检查 USB-485 转换器是否正常工作，或尝试模块 485 AB 之间接 120 欧电阻。

详细的测试演示可参考 [“用户测试文档”](#)

5.6.4 脚本编程案例

(1) 脚本案例 1:

参数名称	参数值
<基本参数>	
Modbus地址	1
通信模块波特率	9600
通信模奇偶校验	8N1
用户脚本1	@DO1=DI1
用户脚本2	@DO2=!DI1

- @DO1=DI1 D01 受控于 DI1 的状态，当 DI1 接通时，D01 接通，反之断开。
- @DO2=!DI1 D02 受控于 DI1 的状态，当 DI1 接通时，D02 断开，反之接通。

(2) 脚本案例 2:

参数名称	参数值
<基本参数>	
Modbus地址	1
通信模块波特率	9600
通信模奇偶校验	8N1
用户脚本1	@DO1=1@D=1@DO1=0@D=1
用户脚本2	@DO2=1@D=2@DO2=0@D=2

- @DO1=1@D=1@DO1=0@D=1 D01 接通 1 秒，然后断开 1 秒，重复执行
- @DO2=1@D=2@DO2=0@D=2 D02 接通 2 秒，然后断开 2 秒，重复执行

(3) 脚本案例 3:

参数名称	参数值
<基本参数>	
Modbus地址	1
通信模块波特率	9600
通信模奇偶校验	8N1
用户脚本1	@IF=(RD=0101)@DO1=1@ENDIF@IF=(RD=0100)@DO1=0@ENDIF@D=0.1
用户脚本2	@IF=(RD=0201)@DO2=1@ENDIF@IF=(RD=0200)@DO2=0@ENDIF@D=0.1

- @IF=(RD=0101)@DO1=1@ENDIF@IF=(RD=0100)@DO1=0@ENDIF@D=0.1
当串口收到 0101 的数据包，D01 导通，当串口收到 0100 的数据包，D01 断开
- @IF=(RD=0201)@DO2=1@ENDIF@IF=(RD=0200)@DO2=0@ENDIF@D=0.1
当串口收到 0201 的数据包，D02 导通，当串口收到 0200 的数据包，D02 断开

(4) 脚本案例 4:

参数名称	参数值
<基本参数>	
Modbus地址	1
通信模块波特率	9600
通信模奇偶校验	8N1
用户脚本1	@IF=DI1~@C=01@ENDIF@D=0.1
用户脚本2	@IF=DI2~@C=02@ENDIF@D=0.1

- @IF=DI1~@C=01@ENDIF@D=0.1
- @IF=DI2~@C=02@ENDIF@D=0.1

当 DI1 变化时，串口发送 01 的数据包，当 DI2 变化时，串口发送 02 的数据

您也可以访问 <http://www.iotcd.cn:8090/RTU.HTML> 网址，实现图形块方式编程，并产生脚本内容。

图形编程示例 1:

- @D01=DI1 D01 受控于 DI1 的状态，当 DI1 接通时，D01 接通，反之断开。
- @D02=!DI1 D02 受控于 DI1 的状态，当 DI1 接通时，D02 断开，反之接通。

[图形化编程](#) > [RTU](#) > 代码

块

程序流程控制

RTU行为控制

变量控制

插入下行内容

DO输出控制

RTU脚本

@DO 控制DO数字输出 DO1 , DI关系 DI1 相同

@DO 控制DO数字输出 DO2 , DI关系 DI1 相反

[图形化编程](#) > [RTU](#) > 代码

块

RTU脚本

@D01=DI1
@D02=!DI1

图形编程示例 2:

- @D01=1@D=1@D01=0@D=1: D01 接通 1 秒，然后断开 1 秒，重复执行
- @D02=1@D=2@D02=0@D=2: D02 接通 2 秒，然后断开 2 秒，重复执行

图形化编程 > RTU > 代码

块

程序流程控制

RTU行为控制

变量控制

插入下行内容

DO输出控制

RTU脚本

@DO 控制DO数字输出 DO1 , 开关选择 开

@D 延时 1

@DO 控制DO数字输出 DO1 , 开关选择 关

@D 延时 1

@DO 控制DO数字输出 DO2 , 开关选择 开

@D 延时 2

@DO 控制DO数字输出 DO2 , 开关选择 关

@D 延时 2

图形化编程 > RTU > 代码

块

RTU脚本

@D01=1@D=1@D01=0@D=1
@D02=1@D=2@D02=0@D=2

图形编程示例 3:

- @IF=(RD=0101)@DO1=1@ENDIF@IF=(RD=0100)@DO1=0@ENDIF@D=0.1
当串口收到 0101 的数据包，D01 导通，当串口收到 0100 的数据包，D01 断开
- @IF=(RD=0201)@DO2=1@ENDIF@IF=(RD=0200)@DO2=0@ENDIF@D=0.1
当串口收到 0201 的数据包，D02 导通，当串口收到 0200 的数据包，D02 断开

图形化编程 > RTU > 代码

块

RTU脚本

程序流程控制

RTU行为控制

变量控制

插入下行内容

DO输出控制

@IF 如果(串口) 串口1 收到特定数据 0101 :

@DO 控制DO数字输出 DO1 , 开关选择 开

@ELSE 否则:

@IF 如果(串口) 串口1 收到特定数据 0100 :

@DO 控制DO数字输出 DO1 , 开关选择 关

@ELSE 否则:

@D 延时 0.1

@IF 如果(串口) 串口1 收到特定数据 0201 :

@DO 控制DO数字输出 DO2 , 开关选择 开

@ELSE 否则:

@IF 如果(串口) 串口1 收到特定数据 0200 :

@DO 控制DO数字输出 DO2 , 开关选择 关

@ELSE 否则:

@D 延时 0.1

图形化编程 > RTU > 代码

块	RTU脚本
---	-------

```
@IF=(RD=0101)@DO1=1@ENDIF@IF=(RD=0100)@DO1=0@ENDIF@D=0.1
@IF=(RD=0201)@DO2=1@ENDIF@IF=(RD=0200)@DO2=0@ENDIF@D=0.1
```

图形编程示例 4:

- @IF=DI1~@C=01@ENDIF@D=0.1
 - @IF=DI2~@C=02@ENDIF@D=0.1
- 当 DI1 变化时，串口发送 01 的数据包，当 DI2 变化时，串口发送 02 的数据

[图形化编程](#) > [RTU](#) > 代码

块

RTU脚本

程序流程控制

RTU行为控制

变量控制

插入下行内容

DO输出控制

@IF 如果(条件) DI1 发生变化

@C 执行HEX数据命令 01 , 附加校验 无

@ELSE 否则:

@D 延时 1

@IF 如果(条件) DI2 发生变化

@C 执行HEX数据命令 02 , 附加校验 无

@ELSE 否则:

@D 延时 1

[图形化编程](#) > [RTU](#) > 代码

块

RTU脚本

@IF=DI1~@C=01@ENDIF@D=1
@IF=DI2~@C=02@ENDIF@D=1

更多完整的脚本编程指令说明，参见《RTU-YX 脚本编程手册》。

六、协议详解

地址域	功能码	数据	差错检验
-----	-----	----	------

Modbus 使用“big-Endian”（大端模式）表示地址和数据项，这就意味着当发射多个字节时，首先发送最高字节。

例如：寄存器地址为 0x0018，首先发送的是 0x00，然后才是 0x14。

一个正常的 Modbus 响应：响应功能码=请求功能码。

一个 Modbus 的异常响应：响应功能码=请求功能码+0x80，提供一个异常码来指示差错原因。

6.1 功能码描述

6.1.1 01 读线圈

可以使用此功能码读取继电器 DOx 的状态。

请求 PDU 详细说明了起始地址，即指定第一个线圈的地址和线圈数量，从零开始寻址线圈，因此寻址线圈 1-N 为 0-(N-1)。

响应 PDU 中 N 个字节的线圈状态的每一个 bit 位代表一个线圈的状态，状态 1=ON，0=OFF。第一个字节的最低位 LSB 代表第 0 号线圈的状态（即起始地址指定的线圈号为 0 号线圈），其他线圈依次类推，一直到这个字节的最高位 MSB 为止，并且后续字节中都是由低到高代表连续的各线圈状态。

如果线圈数量不是 8 的倍数，将用零填充剩余最后数据字节中的剩余比特，字节数量域说明了数据的完整字节数。

请求 PDU

地址	1 个字节	
功能码	1 个字节	0x01
起始地址	2 个字节	0x0018 至 0x001F
线圈数量	2 个字节	n(1 至 n-1)
CRC 校验	2 个字节	

注：线圈状态的字节数 N=线圈数量 n/8，如果余数不等于 0，则 N=n/8+1

错误响应 PDU

地址	1 个字节	
功能码	1 个字节	0x81 （请求功能码+0x80）
异常码	1 个字节	0x01 或 0x02 或 0x03 或 0x04
CRC 校验	2 个字节	

这是一个读离散量 D01 的实例

请求		响应	
地址	01	地址	01
功能码	01	功能码	01
起始地址高 H	00	字节数	01
起始地址低 L	14	D01-D04 状态	01

线圈数量高 H	00	CRC 校验高 H	90
线圈数量低 L	01	CRC 校验低 L	48
CRC 校验高 H	BD		
CRC 校验低 L	CE		

发送：010100140001BDCE RTU 响应：010101019048

D01 的状态字节为 0D，二进制 00000001，D01 是这个字节的 LSB(第 0 位)为 1 表示闭合，其他 D0x 是第(x-1)位为 0 表示断开，用 0 填充未使用位。

6.1.2 03 读保持寄存器/04 读输入寄存器

使用该功能码可以读取所有寄存器包括 AIx、DOx、DIx 的状态。

请求 PDU 详细说明了起始寄存器地址和寄存器数量，从 0 开始寻址寄存器，因此寻址寄存器 1-N 为 0-(N-1)。

响应报文中的寄存器数据每个寄存器有 2 个字节，对于每一个寄存器，第一个字节代表寄存器值的高位，第二个字节代表寄存器值的低位。字节数为寄存器数量乘以 2。对于 AI，一个通道占用 2 个寄存器，4 个字节的值使用浮点数表示，对于 DOx，2 个字节的值 0000 代表继电器断开，0001 代表继电器闭合，对于 DIx，2 个字节的值 0000 代表开关量无输入，0001 代表有输入。

请求 PDU

地址	1 个字节	
功能码	1 个字节	0x03 或 04
起始地址	2 个字节	0x0000 至 0x0017
寄存器数量	2 个字节	n(1 至 N)
CRC 校验	2 个字节	

响应 PDU

地址	1 个字节	
功能码	1 个字节	0x03 或 0x04
字节数	1 个字节	N=2*n
寄存器值	N 个字节	N=2*n, n 为寄存器数量
CRC 校验	2 个字节	

错误响应 PDU

地址	1 个字节	
功能码	1 个字节	0x83 或 0x84 (请求功能码+0x80)
异常码	1 个字节	0x01 或 0x02 或 0x03 或 0x04
CRC 校验	2 个字节	

这是一个读模拟量输入 AI1 的实例

请求		响应	
地址	01	地址	01
功能码	03	功能码	03

起始地址高 H	00	字节数	04
起始地址低 L	00	AI1 值	4 个字节
寄存器数量高 H	00	CRC 校验高 H	
寄存器数量低 L	02	CRC 校验低 L	
CRC 校验高 H	C4		
CRC 校验低 L	0B		

发送: 010300000002C40B RTU 响应:0103044019999AD40F

6.1.3 05 写单个线圈

可以使用该功能码写单个继电器 DOx 为断开或闭合

请求数据域中的常量说明请求的 ON/OFF 状态，十六进制值 0xFF00 请求输出为 ON(闭合)，十六进制值 0x0000 请求输出为 OFF(断开)，其他所有值都是非法的，对输出不起作用，RTU 返回错误响应。

请求域中的输出地址规定了要写入线圈的地址。

正常响应是请求的应答，在写入线圈状态后返回这个正常响应。

请求 PDU

地址	1 个字节	
功能码	1 个字节	0x05
输出地址	2 个字节	0x0014 至 0x0015
输出值	2 个字节	0x0000 或 0xFF00
CRC 校验	2 个字节	

响应 PDU

地址	1 个字节	
功能码	1 个字节	0x05
输出地址	2 个字节	0x0014 至 0x0015
输出值	2 个字节	0x0000 或 0xFF00
CRC 校验	2 个字节	

错误响应 PDU

地址	1 个字节	
功能码	1 个字节	0x85 (请求功能码+0x80)
异常码	1 个字节	0x01 或 0x02 或 0x03 或 0x04
CRC 校验	2 个字节	

这是一个请求写线圈 DO1 为 ON(闭合)的实例

请求		响应	
地址	01	地址	01
功能码	05	功能码	05
输出地址高 H	00	输出地址高 H	00
输出地址低 L	14	输出地址低 L	14

输出值高 H	FF	输出值高 H	FF
输出值低 L	00	输出值低 L	00
CRC 校验高 H	CC	CRC 校验高 H	CC
CRC 校验低 L	3E	CRC 校验低 L	3E

发送：01050014FF00CC3E

RTU 响应：01050014FF00CC3E

6.1.4 06 写单个寄存器

可以使用该功能码写单个继电器 DOx 为断开或闭合。

请求数据域中的寄存器值说明请求的 ON/OFF 状态，十六进制值 0001 请求输出为 ON(闭合)，十六进制值 0x0000 请求输出为 OFF(断开)。

请求域中的寄存器地址规定了要写入线圈的地址。

正常响应是请求的应答，在写入线圈状态后返回这个正常响应。

请求 PDU

地址	1 个字节	
功能码	1 个字节	0x06
寄存器地址	2 个字节	0x0014 至 0x0015
寄存器值	2 个字节	0x0000 至 0xFFFF
CRC 校验	2 个字节	

响应 PDU

地址	1 个字节	
功能码	1 个字节	0x06
寄存器地址	2 个字节	0x0014 至 0x0015
寄存器值	2 个字节	0x0000 至 0xFFFF
CRC 校验	2 个字节	

错误响应 PDU

地址	1 个字节	
功能码	1 个字节	0x86 (请求功能码+0x80)
异常码	1 个字节	0x01 或 0x02 或 0x03 或 0x04
CRC 校验	2 个字节	

这是一个请求写线圈 DO1 为 ON(闭合)的实例

请求		响应	
地址	01	地址	01
功能码	06	功能码	06
寄存器地址高 H	00	寄存器地址高 H	00
寄存器地址低 L	14	寄存器地址低 L	14
寄存器值高 H	00	寄存器值高 H	00
寄存器值低 L	01	寄存器值低 L	01
CRC 校验高 H	08	CRC 校验高 H	08
CRC 校验低 L	0E	CRC 校验低 L	0E

发送：010600140001080E

RTU 响应：010600140001080E

6.1.5 0F 写多个线圈

可以使用此功能码写多个继电器 DOx 为断开或闭合。

请求 PDU 详细说明了起始地址，即指定第一个线圈的地址和线圈数量，从零开始寻址线圈，因此寻址线圈 1-N 为 0-(N-1)。

请求数据域中的内容说明了被请求的 ON/OFF 状态，域比特位中的逻辑“1”请求相应输出为 ON，域比特位中的逻辑“0”请求相应输出为 OFF。从数据域中第一个字节的 bit0 开始到 bit7，然后到第二个字节的 bit0，依次表示第一个线圈到第 n 个线圈的 ON/OFF 值。

正常响应返回功能码、起始地址和线圈数量。

请求 PDU

地址	1 个字节	
功能码	1 个字节	0x0F
起始地址	2 个字节	0x0014 至 0x0015
线圈数量	2 个字节	n (1 至 N)
字节数	1 个字节	$N=n/8$, 或 $N=n/8+1$
输出值	N 个字节	
CRC 校验	2 个字节	

注：线圈输出字节数 $N=$ 线圈数量 $n/8$ ，如果余数不等于 0，则 $N=n/8+1$

响应 PDU

地址	1 个字节	
功能码	1 个字节	0x0F
起始地址	2 个字节	0x0014 至 0x0015
线圈数量	2 个字节	n (1 至 2)
CRC 校验	2 个字节	

错误响应 PDU

地址	1 个字节	
功能码	1 个字节	0x8F (请求功能码+0x80)
异常码	1 个字节	0x01 或 0x02 或 0x03 或 0x04
CRC 校验	2 个字节	

这是一个请求从线圈 DO1 开始写入 1 个线圈的实例

请求		响应	
地址	01	地址	01
功能码	0F	功能码	0F
起始地址高 H	00	起始地址高 H	00
起始地址低 L	14	起始地址低 L	14
线圈数量高 H	00	线圈数量高 H	00
线圈数量低 L	01	线圈数量低 L	01

字节数	01	CRC 校验高 H	D4
输出值	01	CRC 校验低 L	0F
CRC 校验高 H	DF		
CRC 校验低 L	54		

发送：010F0014000201012F51

RTU 响应：010F00140001D40F

D01 的输出值为 01，二进制 00000001，D01 是这个字节的 LSB(第 0 位)为 0 表示断开，D0x 是第(x-1)位为 1 表示闭合，用 0 填充剩余未使用位。

6.1.6 10 写多个寄存器

使用该功能码可以写连续寄存器 D0x 的状态。

请求 PDU 详细说明了起始寄存器地址、寄存器数量、字节数和寄存器值，从零开始寻址寄存器，因此寻址寄存器 1-N 为 0-(N-1)。

寄存器数据中每个寄存器有 2 个字节，对于每一个寄存器，第一个字节代表寄存器值的高位，第二个字节代表寄存器值的低位。字节数为寄存器数量乘以 2，2 个字节的值 0000 代表继电器断开，0001 代表继电器闭合。

正常响应返回功能码、起始地址和被写入寄存器的数量。

请求 PDU

地址	1 个字节	
功能码	1 个字节	0x10
起始地址	2 个字节	0x0014 至 0x0015
寄存器数量	2 个字节	n(1 至 N)
字节数	1 个字节	N=2*n
寄存器值	N 个字节	N=2*n, n 为寄存器数量
CRC 校验	2 个字节	

响应 PDU

地址	1 个字节	
功能码	1 个字节	0x10
起始地址	2 个字节	0x0014 至 0x0015
寄存器数量	2 个字节	n(1 至 2)
CRC 校验	2 个字节	

错误响应 PDU

地址	1 个字节	
功能码	1 个字节	0x90 (请求功能码+0x80)
异常码	1 个字节	0x01 或 0x02 或 0x03 或 0x04
CRC 校验	2 个字节	

这是一个控制继电器 D0x 的实例

请求		响应	
地址	01	地址	01

功能码	10	功能码	10
起始地址高 H	00	起始地址高 H	00
起始地址低 L	14	起始地址低 L	14
寄存器数量高 H	00	寄存器数量高 H	00
寄存器数量低 L	01	寄存器数量低 L	01
字节数	02	CRC 校验高 H	41
D01 寄存器值高 H	00	CRC 校验低 L	CD
D01 寄存器值高 L	01		
CRC 校验高 H	64		
CRC 校验低 L	84		

发送：0110001400010200016484

RTU 响应：01100014000141CD

D01 寄存器值为 0001 表示闭合

6.2 错误码描述

错误码含义：当 DTU 收到错误的 Modbus 指令时，会返回功能码为请求功能码+0x80，紧随着一个字节的错误码代表出错原因。

错误码 01：表示不支持的功能码，众山 DTU 支持上述 8 种功能码，除此之外的功能码都会返回错误码为 01 的错误。

错误码 02：表示起始地址不存在或者起始地址加上寄存器数量后的地址不存在。总的来说表示访问的寄存器不存在。

错误码 03：表示寄存器数量不符合规范或者寄存器值非法。

错误码 04：表示读写寄存器错误。

6.3 CRC 校验算法

CRC 即[循环冗余校验码](#)（Cyclic Redundancy Check）：是数据通信领域中最常用的一种查错校验码，其特征是信息字段和校验字段的长度可以任意选定。循环冗余检查（CRC）是一种数据传输检错功能，对数据进行多项式计算，并将得到的结果附在帧的后面，接收设备也执行类似的算法，以保证数据传输的正确性和完整性。

ModbusRTU 的 CRC16 计算初值：0xFFFF

ModbusRTU 的 CRC16 计算多项式 0xA001（二进制：1010 0000 0000 0001）

附 CRC 校验算法代码：

```
uint16_t mb_crc( uint8_t* snd, uint16_t num )
{
    uint8_t CRC_Lb, CRC_Hb;
    uint8_t CRC_L, CRC_H;
    uint16_t crc;

    CRC_H = 0xFF;
```

```

CRC_L = 0xFF;

for ( uint16_t i = 0; i < num; i++ ) {
    CRC_L = CRC_L ^ snd[ i ];
    for ( uint16_t j = 0; j < 8; j++ ) {
        CRC_Lb = CRC_L;
        if ( ( CRC_L & 1 ) == 1 ) {
            CRC_L = ( CRC_L - 1 ) / 2;
            CRC_Lb = CRC_L;
            CRC_Hb = CRC_H;
            if ( ( CRC_H & 1 ) == 1 ) {
                CRC_L = CRC_L + 128;
                CRC_Lb = CRC_L;
                CRC_H = ( CRC_H - 1 ) / 2;
                CRC_Hb = CRC_H;
            } else {
                CRC_H = CRC_H / 2;
                CRC_Hb = CRC_H;
            }
            CRC_L = CRC_L ^ 1;
            CRC_Lb = CRC_L;
            CRC_H = CRC_H ^ 0xA0;
            CRC_Hb = CRC_H;
        } else {
            CRC_L = CRC_L / 2;
            CRC_Lb = CRC_L;
            CRC_Hb = CRC_H;
            if ( ( CRC_H & 1 ) == 1 ) {
                CRC_L = CRC_L + 128;
                CRC_Lb = CRC_L;
                CRC_H = ( CRC_H - 1 ) / 2;
                CRC_Hb = CRC_H;
            } else {
                CRC_H = CRC_H / 2;
                CRC_Hb = CRC_H;
            }
        }
    }
}

crc = CRC_L;
crc <<= 8;
crc |= CRC_H;
return crc;

```

七、更改记录

v1.6

- * 新增 AI 整数段寄存器读取说明以及示例

v1.5

- * 更新有源、无源接线方式

v1.4

- * 更新外壳尺寸图
- * 更新电池检测接线方式

v1.3

- * 接线图新增三线制

v1.2

- * 修改接线图，调整文字字体

v1.1

- * 修改 3.1 接线图，3.2 跳线图，3.3LED 指示灯配图

v1.0

- * 第一版编写